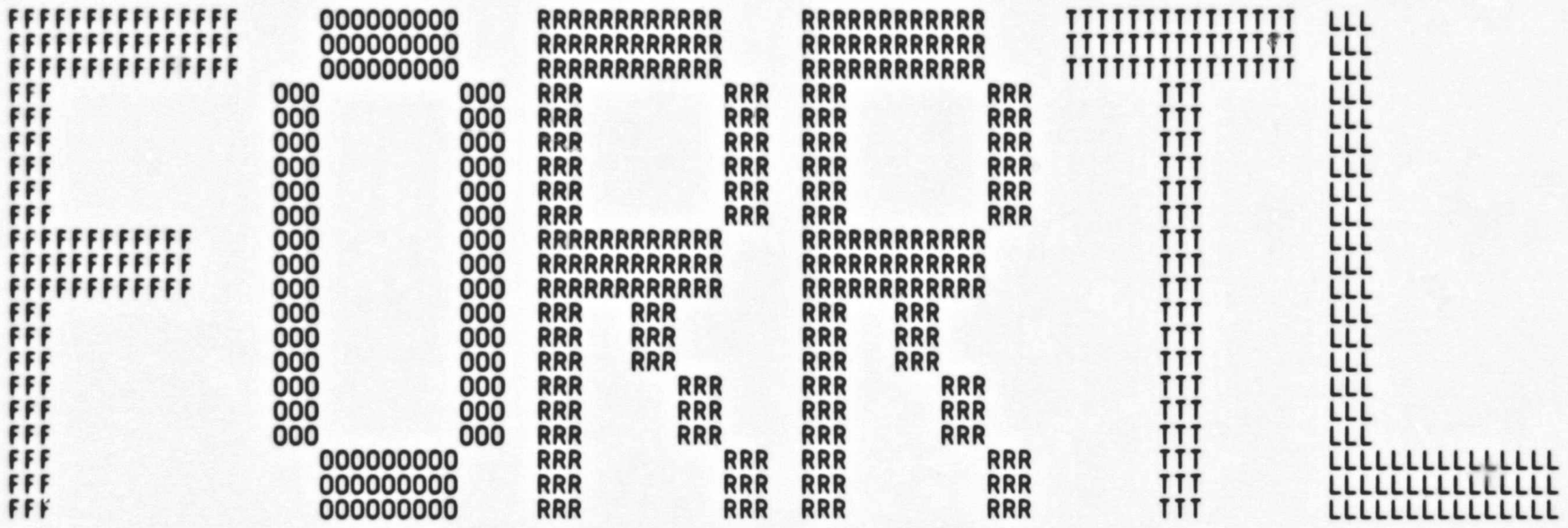
FORTRAN
LLLLLLLLLLLLLLL

```
FFFFFFFFFF    000000    RRRRRRRR    EEEEEEEEEE  RRRRRRRR    RRRRRRRR      SSSSSSSS  NN      NN    SSSSSSSS
FFFFFFFFFF    000000    RRRRRRRR    EEEEEEEEEE  RRRRRRRR    RRRRRRRR      SSSSSSSS  NN      NN    SSSSSSSS
FF          00      00  RR      RR  EE          RR      RR  RR      RR  SS            NN      NN  SS
FF          00      00  RR      RR  EE          RR      RR  RR      RR  SS            NN      NN  SS
FF          00      00  RR      RR  EE          RR      RR  RR      RR  SS            NNNN    NN  SS
FF          00      00  RR      RR  EE          RR      RR  RR      RR  SS            NNNN    NN  SS
FFFFFFFF    00      00  RRRRRRRR    EEEEEEEE    RRRRRRRR    RRRRRRRR      SSSSSS    NN  NN  NN    SSSSSS
FFFFFFFF    00      00  RRRRRRRR    EEEEEEEE    RRRRRRRR    RRRRRRRR      SSSSSS    NN  NN  NN    SSSSSS
FF          00      00  RR   RR     EE          RR   RR     RR   RR                 SS  NN    NNNN          SS
FF          00      00  RR   RR     EE          RR   RR     RR   RR                 SS  NN    NNNN          SS
FF          00      00  RR    RR    EE          RR    RR    RR    RR               SS  NN      NN          SS      ....
FF          00      00  RR    RR    EE          RR    RR    RR    RR               SS  NN      NN          SS      ....
FF            000000    RR      RR  EEEEEEEEEE  RR      RR  RR      RR  SSSSSSSS  NN      NN  SSSSSSSS        ....
FF            000000    RR      RR  EEEEEEEEEE  RR      RR  RR      RR  SSSSSSSS  NN      NN  SSSSSSSS        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE FOR$ERRSNS (%TITLE'FORTRAN ERRSNS and save error info'
;     2          0002  0         IDENT = '1-003'          ! File: FORERRSNS.B32  Edit: SBL1003
;     3          0003  0         ) =
;     4          0004  1 BEGIN
;     5          0005  1
;     6          0006  1 !
;     7          0007  1 !*****************************************************************************
;     8          0008  1 !*                                                                           *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                     *
;    12          0012  1 !*                                                                           *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    18          0018  1 !*  TRANSFERRED.                                                             *
;    19          0019  1 !*                                                                           *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    22          0022  1 !*  CORPORATION.                                                             *
;    23          0023  1 !*                                                                           *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*****************************************************************************
;    29          0029  1 !
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 ! FACILITY:  FORTRAN Support Library
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !
;    37          0037  1 !       Return information about last FORTRAN error (ERRSNS).
;    38          0038  1 !       Also an internal routine to save that information when
;    39          0039  1 !       an error occurs in OWN storage.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  User Mode - AST re-entrant
;    42          0042  1 !
;    43          0043  1 ! AUTHOR:  Thomas N. Hastings, CREATION DATE:  8-Nov-1977
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 !       Thomas N. Hastings, 8-Nov-1977: VERSION 0
;    48          0048  1 ! [Previous edit history removed.  SBL 29-Jun-1983]
;    49          0049  1 ! 1-001 - Update copyright statement and version number.  JBS 16-NOV-78
;    50          0050  1 ! 1-002 - Declare NULLPARAMETER and ACTUALPARAMETER for new BLISS
;    51          0051  1 !         compiler.  JBS 22-NOV-78
;    52          0052  1 ! 1-003 - Use prologue file.  Add FOR$$INIT_ERRSET and call back to
;    53          0053  1 !         routine to store error info in ERRSET/ERRTST table.  SBL 29-Jun-1983
;    54          0054  1 !--
```

```
;    56          0055  1 !
;    57          0056  1 ! PROLOGUE FILE:
;    58          0057  1 !
;    59          0058  1
;    60          0059  1 REQUIRE 'RTLIN:FORPROLOG';              ! FOR$ definitions
;    61          0125  1
;    62          0126  1 !
;    63          0127  1 ! TABLE OF CONTENTS:
;    64          0128  1 !
;    65          0129  1
;    66          0130  1 FORWARD ROUTINE
;    67          0131  1         FOR$ERRSNS: NOVALUE,            ! Return error information about last error
;    68          0132  1         FOR$ERRSNS_W: NOVALUE,          ! Same except word size args
;    69          0133  1         FOR$$ERRSNS_SAV: NOVALUE,       ! Internal routine to save information
;    70          0134  1         FOR$$INIT_ERRSET: NOVALUE;      ! Initialize for ERRSET/ERRTST
;    71          0135  1
;    72          0136  1 !
;    73          0137  1 ! MACROS:
;    74          0138  1 !
;    75          0139  1
;    76          0140  1 !
;    77          0141  1 ! EQUATED SYMBOLS:
;    78          0142  1 !
;    79          0143  1 !       The following offsets are used to access OWN vectors, LOCAL vectors, and formals:
;    80          0144  1 !
;    81          0145  1 LITERAL
;    82          0146  1         NPARMAX = 5;                    ! No. of parameters max.
;    83          0147  1
;    84          0148  1 !
;    85          0149  1 ! OWN STORAGE:
;    86          0150  1 !
;    87          0151  1 OWN
;    88          0152  1         LAST_ERROR_INFO: VECTOR [NPARMAX+1],    ! Last error info (non-AST level)
;    89          0153  1         LAST_AST_INFO: VECTOR [NPARMAX+1],      ! Last error info (AST level)
;    90          0154  1                                                 ! 0th entry not used
;    91          0155  1         RECORD_ERROR_ADDR: INITIAL (0);         ! Address of RECORD_ERROR rtn
;    92          0156  1
;    93          0157  1 !
;    94          0158  1 ! EXTERNAL REFERENCES:
;    95          0159  1 !
;    96          0160  1
;    97          0161  1 EXTERNAL ROUTINE
;    98          0162  1         LIB$AST_IN_PROG: ADDRESSING_MODE (GENERAL);     ! TRUE if AST in progress
```

```
;  100     0163  1 %SBTTL'FOR$ERRSNS'
;  101     0164  1 GLOBAL ROUTINE FOR$ERRSNS (
;  102     0165  1         FORT_ERR_NO,            ! Optional adr. to get FORTRAN error no.
;  103     0166  1         RMS_STS,                ! Optional adr. ro get RMS status
;  104     0167  1         RMS_STV,                ! Optional adr. to get RMS/VMS value
;  105     0168  1         FORT_LUN,               ! Optional adr. to get FORTRAN logical unit number
;  106     0169  1         VAX_T1_COND_VAL)        ! Optional adr. to get VAX-11 condition value
;  107     0170  1         :NOVALUE =                      ! No value returned
;  108     0171  1 !++
;  109     0172  1 ! FUNCTIONAL DESCRIPTION:
;  110     0173  1 !
;  111     0174  1 !       Returns information about last FORTRAN error, if any, and clears it to 0
;  112     0175  1 !       Separate data bases are kept for AST and non-AST level
;  113     0176  1 !       so that they do not conflict.  That is the last error returned
;  114     0177  1 !       is that which occurred at the level of the caller (AST level
;  115     0178  1 !       or not AST level).  Therefore a modular re-entrant procedure
;  116     0179  1 !       can CALL FOR$ERRSNS and still remain AST re-entrant
;  117     0180  1 !       provided that FOR$ERRSNS is called after an I/O statement
;  118     0181  1 !       with no intervening calls (which might call FOR$ERRSNS).
;  119     0182  1 !       All parameters are optional.  A call with no parameters
;  120     0183  1 !       has the effect of clearing the error information at the current level.
;  121     0184  1 !       It is good practice to CALL FOR$ERRSNS before doing
;  122     0185  1 !       I/O to initialize the OWN storage.  Otherwise a previous
;  123     0186  1 !       I/O error may have left error information in the OWN storage.
;  124     0187  1 !       Note: successful I/O operations do not affect the
;  125     0188  1 !       error information OWN storage.
;  126     0189  1 !
;  127     0190  1 ! FORMAL PARAMETERS:
;  128     0191  1 !
;  129     0192  1 !       [FORT_ERR_NO.wlu.r      Optional adr. to receive FORTRAN error no.
;  130     0193  1 !       [RMS_STS.wlu.r          Optional adr. to receive RMS status
;  131     0194  1 !       [RMS_STV.wlu.r          Optional adr. to receive RMS/VMS value
;  132     0195  1 !       [FORT_LUN.wlu.r Optional adr. to receive FORTRAN logical unit no.
;  133     0196  1 !       [VAX_T1_COND_VAL.wlc.r]]]]] Optional adr. to receive VAX-11 condition value
;  134     0197  1 !
;  135     0198  1 ! IMPLICIT INPUTS:
;  136     0199  1 !
;  137     0200  1 !       VMS info whether an AST is in progress or not (using LIB$AST_IN_PROG)
;  138     0201  1 !       Either LAST_ERROR_INFO vector or LAST_AST_INFO vector
;  139     0202  1 !
;  140     0203  1 ! IMPLICIT OUTPUTS:
;  141     0204  1 !
;  142     0205  1 !       Either LAST_ERROR_INFO vector or LAST_AST_INFO vector is cleared
;  143     0206  1 !       depending on the level of the caller (AST or in progress or not)
;  144     0207  1 !
;  145     0208  1 ! ROUTINE VALUE:
;  146     0209  1 ! COMPLETION CODES:
;  147     0210  1 !
;  148     0211  1 !       NONE
;  149     0212  1 !
;  150     0213  1 ! SIDE EFFECTS:
;  151     0214  1 !
;  152     0215  1 !       The OWN storage used to remember the last error is set to 0.
;  153     0216  1 !--
;  154     0217  1
;  155     0218  2     BEGIN
;  156     0219  2
```

```
;  157          0220  2     BUILTIN NULLPARAMETER, ACTUALPARAMETER;
;  158          0221  2
;  159          0222  2     LOCAL
;  160          0223  2         LAST_INFO: REF VECTOR[NPARMAX+1,LONG];  ! Base pointer to either LAST_ERROR_INFO or LAST_AST_INFO
;  161          0224  2
;  162          0225  2
;  163          0226  2     !+
;  164          0227  2     ! Determine whether an AST is in progress or not and set up
;  165          0228  2     ! base pointer LAST_INFO to point to OWN storage for that level.
;  166          0229  2     !-
;  167          0230  2
;  168          0231  2     LAST_INFO = (IF LIB$AST_IN_PROG () THEN LAST_AST_INFO ELSE LAST_ERROR_INFO);
;  169          0232  2
;  170          0233  2     !+
;  171          0234  2     ! Copy appropriate OWN storage to the caller's parameters
;  172          0235  2     ! which are present and clear each OWN storage location.
;  173          0236  2     !-
;  174          0237  2
;  175          0238  2     INCR I FROM 1 TO NPARMAX DO
;  176          0239  3         BEGIN
;  177          0240  3         IF NOT NULLPARAMETER (.I) THEN ACTUALPARAMETER(.I) = .LAST_INFO[.I];
;  178          0241  3         LAST_INFO[.I] = 0;
;  179          0242  2         END;
;  180          0243  2
;  181          0244  2     !+
;  182          0245  2     ! Return
;  183          0246  2     !-
;  184          0247  2
;  185          0248  2     RETURN
;  186          0249  1     END;


                                                    .TITLE  FOR$ERRSNS FORTRAN ERRSNS and save error info
                                                    .IDENT  \1-003\

                                                    .PSECT  _FOR$DATA,NOEXE,  PIC,2

                                            00000 LAST_ERROR_INFO:
                                                    .BLKB   24
                                            00018 LAST_AST_INFO:
                                                    .BLKB   24
                                   00000000 00030 RECORD_ERROR_ADDR:
                                                    .LONG   0                                                       ;

                                                    .EXTRN  LIB$AST_IN_PROG

                                                    .PSECT  _FOR$CODE,NOWRT,  SHR,  PIC,2

                                              0004 00000            .ENTRY  FOR$ERRSNS, Save R2                     ; 0164
                00000000G   00                00  FB 00002            CALLS   #0, LIB$AST_IN_PROG                 ; 0231
                            09                50  E9 00009            BLBC    R0, 1$                              ;
                            52 00000000'      EF  9E 0000C            MOVAB   LAST_AST_INFO, LAST_INFO            ;
                                              07  11 00013            BRB     2$                                  ;
                            52 00000000'      EF  9E 00015 1$:        MOVAB   LAST_ERROR_INFO, LAST_INFO          ;
                            50                01  D0 0001C 2$:        MOVL    #1, I                               ; 0241
          50                6C              08  ED 0001F 3$:        CMPZV   #0, #8, (AP), I                     ; 0240
                                              0D  19 00024            BLSS    4$                                  ;
```

```
                                6C40  D5 00026          TSTL    (AP)[I]                                 ;
                                  08  13 00029          BEQL    4$                                      ;
                          51      6C40  D0 0002B          MOVL    (AP)[I], R1                             ;
                          61      6240  D0 0002F          MOVL    (LAST_INFO)[I], (R1)                    ;
                                  6240  D4 00033 4$:      CLRL    (LAST_INFO)[I]                          ; 0241
                      E5  50        05  F3 00036          AOBLEQ  #5, I, 3$                               ; 0238
                                        04 0003A          RET                                             ; 0249

; Routine Size:  59 bytes,    Routine Base:  _FOR$CODE + 0000
```

```
;  188     0250  1 %SBTTL'FOR$ERRSNS_W'
;  189     0251  1 GLOBAL ROUTINE FOR$ERRSNS_W (
;  190     0252  1         FORT_ERR_NO,                ! Optional adr. to get FORTRAN error no.
;  191     0253  1         RMS_STS,                    ! Optional adr. ro get RMS status
;  192     0254  1         RMS_STV,                    ! Optional adr. to get RMS/VMS value
;  193     0255  1         FORT_LUN,                   ! Optional adr. to get FORTRAN logical unit number
;  194     0256  1         VAX_T1_COND_VAL)            ! Optional adr. to get VAX-11 condition value
;  195     0257  1         :NOVALUE =                          ! No value returned
;  196     0258  1 !++
;  197     0259  1 ! FUNCTIONAL DESCRIPTION:
;  198     0260  1 !
;  199     0261  1 !       FOR$ERRSNS and FOR$ERRSNS_W are the same routines except
;  200     0262  1 !       for the data size returned.
;  201     0263  1 !       Returns information about last FORTRAN error, if any, and clears it to 0
;  202     0264  1 !       Separate data bases are kept for AST and non-AST level
;  203     0265  1 !       so that they do not conflict.  That is the last error returned
;  204     0266  1 !       is that which occurred at the level of the caller (AST level
;  205     0267  1 !       or not AST level).  Therefore a modular re-entrant procedure
;  206     0268  1 !       can CALL FOR$ERRSNS_W and still remain AST re-entrant
;  207     0269  1 !       provided that FOR$ERRSNS_W is called after an I/O statement
;  208     0270  1 !       with no intervening calls (which might call FOR$ERRSNS_W).
;  209     0271  1 !       All parameters are optional.  A call with no parameters
;  210     0272  1 !       has the effect of clearing the error information at the current level.
;  211     0273  1 !       It is good practice to CALL FOR$ERRSNS_W before doing
;  212     0274  1 !       I/O to initialize the OWN storage.  Otherwise a previous
;  213     0275  1 !       I/O error may have left error information in the OWN storage.
;  214     0276  1 !       Note: successful I/O operations do not affect the
;  215     0277  1 !       error information OWN storage.
;  216     0278  1 !
;  217     0279  1 ! FORMAL PARAMETERS:
;  218     0280  1 !
;  219     0281  1 !       [FORT_ERR_NO.wwu.r      Optional adr. to receive FORTRAN error no.
;  220     0282  1 !       [RMS_STS.wwu.r          Optional adr. to receive RMS status
;  221     0283  1 !       [RMS_STV.wwu.r          Optional adr. to receive RMS/VMS value
;  222     0284  1 !       [FORT_LUN.wwu.r         Optional adr. to receive FORTRAN logical unit no.
;  223     0285  1 !       [VAX_T1_COND_VAL.wwu.r]]]]] Optional adr. to receive VAX-11 condition value<15:0>
;  224     0286  1 !
;  225     0287  1 ! IMPLICIT INPUTS:
;  226     0288  1 !
;  227     0289  1 !       VMS info whether an AST is in progress or not (using LIB$AST_IN_PROG)
;  228     0290  1 !       Either LAST_ERROR_INFO vector or LAST_AST_INFO vector
;  229     0291  1 !
;  230     0292  1 ! IMPLICIT OUTPUTS:
;  231     0293  1 !
;  232     0294  1 !       Either LAST_ERROR_INFO vector or LAST_AST_INFO vector is cleared
;  233     0295  1 !       depending on the level of the caller (AST or in progress or not)
;  234     0296  1 !
;  235     0297  1 ! ROUTINE VALUE:
;  236     0298  1 ! COMPLETION CODES:
;  237     0299  1 !
;  238     0300  1 !       NONE
;  239     0301  1 !
;  240     0302  1 ! SIDE EFFECTS:
;  241     0303  1 !
;  242     0304  1 !       The OWN storage used to remember the last error is set to 0.
;  243     0305  1 !--
;  244     0306  1
```

```
;  245          0307  2     BEGIN
;  246          0308  2
;  247          0309  2     BUILTIN NULLPARAMETER, ACTUALPARAMETER;
;  248          0310  2
;  249          0311  2     LOCAL
;  250          0312  2         NPARMAX_LONGS: VECTOR[NPARMAX,LONG];    ! Five longwords to contain results from FOR$ERRSNS
;  251          0313  2
;  252          0314  2
;  253          0315  2     !+
;  254          0316  2     ! Get ERRSNS data into longword LOCAL storage
;  255          0317  2     !-
;  256          0318  2
;  257          0319  2     FOR$ERRSNS (NPARMAX_LONGS[0], NPARMAX_LONGS[1], NPARMAX_LONGS[2],
;  258          0320  2         NPARMAX_LONGS[3], NPARMAX_LONGS[4]);
;  259          0321  2
;  260          0322  2     !+
;  261          0323  2     ! Copy longwords back to caller's word parameters
;  262          0324  2     ! Parameters are 1-origin, VECTOR is 0-origin.
;  263          0325  2     !-
;  264          0326  2
;  265          0327  2     INCR I FROM 1 TO NPARMAX DO
;  266          0328  2         IF NOT NULLPARAMETER (.I) THEN ACTUALPARAMETER(.I)<0,16> = .NPARMAX_LONGS[.I-1];
;  267          0329  2
;  268          0330  2     !+
;  269          0331  2     ! Return
;  270          0332  2     !-
;  271          0333  2
;  272          0334  2     RETURN
;  273          0335  1     END;


                                    0000 00000          .ENTRY  FOR$ERRSNS_W, Save nothing                  ; 0251
                            5E        14   C2 00002     SUBL2   #20, SP                                     ;
                        10  AE        9F   00005     PUSHAB  NPARMAX_LONGS+16                            ; 0320
                        10  AE        9F   00008     PUSHAB  NPARMAX_LONGS+12                            ;
                        10  AE        9F   0000B     PUSHAB  NPARMAX_LONGS+8                             ; 0319
                        10  AE        9F   0000E     PUSHAB  NPARMAX_LONGS+4                             ;
                        10  AE        9F   00011     PUSHAB  NPARMAX_LONGS                               ;
                    AD  AF            05   FB 00014     CALLS   #5, FOR$ERRSNS                          ;
                        50            01   D0 00018     MOVL    #1, I                                  ; 0327
                50      6C            08        00   ED 0001B 1$:  CMPZV   #0, #8, (AP), I              ; 0328
                                      0E        19   00020     BLSS    2$                              ;
                                  6C40  D5   00022     TSTL    (AP)[I]                                 ;
                                      09        13   00025     BEQL    2$                              ;
                                  6C40  D0   00027     MOVL    (AP)[I], R1                             ;
                                  FC AE40   F7   0002B     CVTLW   NPARMAX_LONGS-4[I], (R1)            ;
                            E7        50        05   F3 00030 2$:  AOBLEQ  #5, I, T$                   ;
                                                   04   00034     RET                                     ; 0335

; Routine Size:  53 bytes,    Routine Base:  _FOR$CODE + 003B
```

```
;  275          0336  1 %SBTTL'FOR$$ERRSNS_SAV'
;  276          0337  1 GLOBAL ROUTINE FOR$$ERRSNS_SAV (
;  277          0338  1         FORT_ERR_NO,              ! FORTRAN error number (0:120) or 32-bit cond value
;  278          0339  1                                   ! for some other facility other than FOR$.
;  279          0340  1         RMS_STS,                  ! value to set RMS status
;  280          0341  1         RMS_STV,                  ! value to set RMS/VMS value
;  281          0342  1         FORT_LUN,                 ! value to set FORTRAN logical unit number
;  282          0343  1         VAX_T1_COND_VAL)          ! value to set VAX-11 condition value
;  283          0344  1         :NOVALUE =                ! No value returned
;  284          0345  1 !++
;  285          0346  1 ! FUNCTIONAL DESCRIPTION:
;  286          0347  1 !
;  287          0348  1 !       Called on every error condition.  Sets FORTRAN error info OWN storage
;  288          0349  1 !       for use in sub-sequent calls by FOR$ERRSNS and FOR$ERRSNS_W.
;  289          0350  1 !       Separate data bases are kept for AST and non-AST level
;  290          0351  1 !       so that they do not conflict.  That is the last error returned
;  291          0352  1 !       is that which occurred at the level of the caller (AST level
;  292          0353  1 !       or not AST level).  Therefore a modular re-entrant procedure
;  293          0354  1 !       can CALL FOR$ERRSNS_SAV and still remain AST re-entrant
;  294          0355  1 !       Note: successful I/O operations do not affect the
;  295          0356  1 !       error information OWN storage.
;  296          0357  1 !       Non-FORTRAN specific errors should be indicated with:
;  297          0358  1 !               FORT_ERR_NO = 32-bit condition value (not FOR$ facility)
;  298          0359  1 !               VAX_T1_COND_VAL = same, ie. OTS$_FATINTERR, OTS$_INTDATCOR
;  299          0360  1 !       In this case, the FORTRAN error number stored will be FOR$K_NOTFORSPE
;  300          0361  1 !       which has a value of 1 and indicated a non-FORTRAN specific error.
;  301          0362  1 !
;  302          0363  1 ! FORMAL PARAMETERS:
;  303          0364  1 !
;  304          0365  1 !       FORT_ERR_NO.rlu.v       value to specify FORTRAN error number (0:120)
;  305          0366  1 !                               or 32-bit condition value for another facility error.
;  306          0367  1 !       RMS_STS.wlu.v           value to set RMS status
;  307          0368  1 !       RMS_STV.wlu.v           value to set RMS/VMS value
;  308          0369  1 !       FORT_LUN.wlu.v          value to set FORTRAN logical unit no.
;  309          0370  1 !       VAX_T1_COND_VAL.wlc.v   value to set VAX-11 condition value
;  310          0371  1 !
;  311          0372  1 ! IMPLICIT INPUTS:
;  312          0373  1 !
;  313          0374  1 !       VMS info whether an AST is in progress or not (using LIB$AST_IN_PROG)
;  314          0375  1 !
;  315          0376  1 ! IMPLICIT OUTPUTS:
;  316          0377  1 !
;  317          0378  1 !       Either LAST_ERROR_INFO vector or LAST_AST_INFO vector is set
;  318          0379  1 !       depending on the level of the caller (AST or in progress or not)
;  319          0380  1 !
;  320          0381  1 ! ROUTINE VALUE:
;  321          0382  1 ! COMPLETION CODES:
;  322          0383  1 !
;  323          0384  1 !       NONE
;  324          0385  1 !
;  325          0386  1 ! SIDE EFFECTS:
;  326          0387  1 !
;  327          0388  1 !       The OWN storage used to remember the last error is set.
;  328          0389  1 !--
;  329          0390  1
;  330          0391  2     BEGIN
;  331          0392  2
```

```
;  332     0393  2     BUILTIN ACTUALPARAMETER;
;  333     0394  2
;  334     0395  2     MAP
;  335     0396  2         VAX_11_COND_VAL : BLOCK[4, BYTE];       ! Condition value
;  336     0397  2     LOCAL
;  337     0398  2         LAST_INFO: REF VECTOR[NPARMAX+1,LONG];  ! Base pointer to either LAST_ERROR_INFO or LAST_AST_INFO
;  338     0399  2
;  339     0400  2
;  340     0401  2     !+
;  341     0402  2     ! Determine whether an AST is in progress or not and set up
;  342     0403  2     ! base pointer LAST_INFO to point to OWN storage for that level.
;  343     0404  2     !-
;  344     0405  2
;  345     0406  2     LAST_INFO = (IF LIB$AST_IN_PROG () THEN LAST_AST_INFO ELSE LAST_ERROR_INFO);
;  346     0407  2
;  347     0408  2     !+
;  348     0409  2     ! Copy all formals to OWN storage
;  349     0410  2     !-
;  350     0411  2
;  351     0412  2     INCR I FROM 1 TO NPARMAX DO
;  352     0413  2         LAST_INFO[.I] = ACTUALPARAMETER(.I);
;  353     0414  2
;  354     0415  2     !+
;  355     0416  2     ! Check FORTRAN error number, if already a 32-bit condition value
;  356     0417  2     ! change to FOR$K_NOTFORSPE to indicate a non-FORTRAN specific error
;  357     0418  2     ! (error number = 1).
;  358     0419  2     !-
;  359     0420  2
;  360     0421  2     IF .FORT_ERR_NO GTRU FOR$K_MAX_ERR
;  361     0422  2     THEN
;  362     0423  2         LAST_INFO[1] = FOR$K_NOTFORSPE;
;  363     0424  2
;  364     0425  2     !+
;  365     0426  2     ! If the user is using ERRSET or ERRTST, call back to RECORD_ERROR
;  366     0427  2     ! in module COM$ERRSET_TST to record the error.
;  367     0428  2     !-
;  368     0429  2
;  369     0430  2     IF .RECORD_ERROR_ADDR NEQA 0
;  370     0431  2     THEN
;  371     0432  2         BLISS (.RECORD_ERROR_ADDR, .LAST_INFO [1]);
;  372     0433  2
;  373     0434  2     !+
;  374     0435  2     ! Return
;  375     0436  2     !-
;  376     0437  2
;  377     0438  2     RETURN
;  378     0439  1     END;


                                    0004 00000          .ENTRY  FOR$$ERRSNS_SAV, Save R2                        ; 0337
                 52 00000000'   EF   9E 00002          MOVAB   LAST_AST_INFO, R2                               ;
00000000G        00             00   FB 00009          CALLS   #0, LIB$AST_IN_PROG                             ; 0406
                 05             50   E9 00010          BLBC    R0, 1$                                          ;
                 51             62   9E 00013          MOVAB   LAST_AST_INFO, LAST_INFO                        ;
```

```
                                        04  11 00016             BRB     2$                                      ;
                                51      E8  A2 9E 00018 1$:      MOVAB   LAST_ERROR_INFO, LAST_INFO              ;
                                50          01  D0 0001C 2$:    MOVL    #1, I                                   ; 0413
                              6140              6C40  D0 0001F 3$:  MOVL  (AP)[I], (LAST_INFO)[I]
                        F7    50          05  F3 00024           AOBLEQ  #5, I, 3$                               ;
                  0000005D    8F      04  AC  D1 00028           CMPL    FORT_ERR_NO, #93                        ; 0421
                                        04  1B 00030             BLEQU   4$                                      ;
                            04  A1          01  D0 00032         MOVL    #1, 4(LAST_INFO)                        ; 0423
                                50      18  A2  D0 00036 4$:     MOVL    RECORD_ERROR_ADDR, R0                   ; 0430
                                        06  13 0003A             BEQL    5$                                      ;
                                    04  A1  DD 0003C             PUSHL   4(LAST_INFO)                            ; 0432
                                60          01  FB 0003F         CALLS   #1, (R0)                                ;
                                            04 00042 5$:         RET                                             ; 0439

; Routine Size:  67 bytes,    Routine Base:  _FOR$CODE + 0070
```

```
;  380          0440  1 %SBTTL'FOR$$INIT_ERRSET'
;  381          0441  1 GLOBAL ROUTINE FOR$$INIT_ERRSET (
;  382          0442  1         ROUTINE_ADDR
;  383          0443  1         ) :NOVALUE =
;  384          0444  1 !++
;  385          0445  1 ! FUNCTIONAL DESCRIPTION:
;  386          0446  1 !
;  387          0447  1 !       Called by COM_STARTUP in COM$$ERRSET_TST when ERRSET or ERRTST
;  388          0448  1 !       has been included in the image.  It passes the address of a routine
;  389          0449  1 !       (RECORD_ERROR) which we are to call whenever an I/O error occurs.
;  390          0450  1 !       This allows the ERRSET/ERRTST table to keep informed of all I/O
;  391          0451  1 !       errors that occur, not just the ones that don't get trapped.
;  392          0452  1 !
;  393          0453  1 ! FORMAL PARAMETERS:
;  394          0454  1 !
;  395          0455  1 !       ROUTINE_ADDR    The address of the routine to call.  This is stored
;  396          0456  1 !                       in our OWN storage RECORD_ERROR_ADDR.
;  397          0457  1 !
;  398          0458  1 ! IMPLICIT INPUTS:
;  399          0459  1 !
;  400          0460  1 !       NONE
;  401          0461  1 !
;  402          0462  1 ! IMPLICIT OUTPUTS:
;  403          0463  1 !
;  404          0464  1 !       The routine's address is stored in RECORD_ERROR_ADDR .
;  405          0465  1 !
;  406          0466  1 ! ROUTINE VALUE:
;  407          0467  1 !
;  408          0468  1 !       NONE
;  409          0469  1 !
;  410          0470  1 ! SIDE EFFECTS:
;  411          0471  1 !
;  412          0472  1 !       NONE
;  413          0473  1 !--
;  414          0474  1
;  415          0475  2     BEGIN
;  416          0476  2
;  417          0477  2     !+
;  418          0478  2     ! Store the routine address.
;  419          0479  2     !-
;  420          0480  2
;  421          0481  2     RECORD_ERROR_ADDR = .ROUTINE_ADDR;
;  422          0482  2
;  423          0483  2     !+
;  424          0484  2     ! Return
;  425          0485  2     !-
;  426          0486  2
;  427          0487  2     RETURN
;  428          0488  1     END;


                                        0000 00000          .ENTRY  FOR$$INIT_ERRSET, Save nothing                  ; 0441
00000000'  EF       04    AC    DO 00002          MOVL    ROUTINE_ADDR, RECORD_ERROR_ADDR                         ; 0481
                                    04 0000A          RET                                                         ; 0488
```

```
; Routine Size:  11 bytes,    Routine Base:  _FOR$CODE + 00B3

;  429          0489  1 END                                ! end of module
;  430          0490  0 ELUDOM



;                                PSECT SUMMARY
;
;      Name                     Bytes                      Attributes
;
;  _FOR$DATA                      52  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _FOR$CODE                     190  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                            Library Statistics
;                                    -------- Symbols --------      Pages       Processing
;      File                          Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1   9776         0         0         581        00:01.0
;  _$255$DUA28:[FORRTL.OBJ]FORLIB.L32;1   711         2         0          52        00:00.6
;  _$255$DUA28:[FORRTL.OBJ]RTLLIB.L32;1    36         0         0           8        00:00.1



;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:FORERRSNS/OBJ=OBJ$:FORERRSNS MSRC$:FORERRSNS/UPDATE=(ENH$:FORERRSNS
;       )

; Size:          190 code + 52 data bytes
; Run Time:        00:07.7
; Elapsed Time:    00:20.1
; Lines/CPU Min:    3798
; Lexemes/CPU-Min:  8945
; Memory Used:  50 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FORFIND
LIS

FORFMTINT
LIS

FOREXITHA
LIS

FORENODEF
LIS

FORENCOMF
LIS

FORIDATE
LIS

FORENDFIL
LIS

FORERRSNS
LIS

FOREXIT
LIS

FORERROR
LIS

FORFMTCP
LIS

FORENCOMO
LIS

FORTNIDES
LIS